This document contains some basics of the complexity theory. It is mostly based on the lecture course delivered at CS dept. by Meran G. Furugyan. Differences are subtle. Disclaimer of warranties apply: if you fail an exam because of an error on my part, it's your problem – you could attend lectures after all. However, major bugs have some chances to be fixed and can be reported to `<ghost@cs.msu.su>`, and new versions will probably appear at `http://chronos.cs.msu.su/~ghost/MyDocuments/Text/complexity.pdf`

Beware! It is very *alpha version.*

Contents

1 NP-complete problems

Due to limited author's time, definitions of Turing machine and NP-completeness are omitted. They can be found in last year's lectures that are available somewhere in the net.

1.1 Basic NP-complete problems

[There should go tree]

Problem 1 (SAT) *Given a CNF $E = E_1 \dots E_n$, find if it's satisfiable*

Problem 2 (SAT-3) *Given a CNF $E = E_1 \dots E_n$, where each E_k has the form $E_k = v_{k1} \vee v_{k2} \vee v_{k3}$ find if it's satisfiable*

Problem 3 (3DM) *Given three sets W, V, Z or equal cardinality N and a set $W \subseteq W \times V \times Z$, find if there's $W' \subseteq W$ such that $|W'| = N$ and no pair of elements in W' agree in any coordinate. That is, each element from W, V and Z occurs in exactly one triad from W'*

Note: This is very similar to 2-dimensional matching: given two equally sized groups of boys and girls, and a set of possible pairs, pair 'em all. Compexities, however, are *very* different.

Problem 4 (Partition) *Given a set $A = (a_1, \dots, a_n)$ of numbers, find partition of A into A' and A'' ($A' \cup A'' = A$, $A' \cap A'' = \emptyset$) such that $\sum_{a_i \in A'} a_i = \sum_{a_i \in A''} a_i$*

Problem 5 (Vertex cover) *Given an undirected graph $G = (V, A)$ and a number K determine if there's $V' \subseteq V$ such that $|V'| \leqslant K$ and $\forall (u, v) \in A \; u \in V' \vee v \in V'$*

Problem 6 (Independent Set) *Given an undirected graph $G = (V, A)$ and a number K determine if there's $V' \subseteq V$ such that $|V'| \geqslant K$ and $\forall u, v \in V'$ u and v are not adjacent in G.*

Problem 7 (Clique) *Given an undirected graph $G = (V, A)$ and a number K determine if there's $V' \subseteq V$ such that $|V'| \geqslant K$ and subgraph induced by V' is complete.*

Problem 8 (Hamiltonian circuit) *Given an undirected graph $G = (V, A)$ find if there's simple circuit passing through all the vertices in the graph.*

Theorem 1 *SAT is in NPC.*

Proof. (The fact that SAT $\in \mathbb{NP}$ is considered obvious) The aim is the prove that $\forall L \in \mathbb{NP}$, $L \leqslant_p L(SAT)$. From the definition of $\mathbb{NP}$ we know that exists NDMT M accepting L, and $\exists p$ such that time bound is $p(n)$, where n is string length. That is, exists a computation of M with less than $p(n)$ that accepts that string. Existence of such computation can be formulated in terms of boolean expressions.

[Definition of NDMT, numbering of states/symbols, &c.]

i. Q_{ik} – does machine in time i in state k.

ii. H_{ij} – does machine in time i look at position j.

iii. S_{ijl} – does machine in time i in position j contain symbol l.

Indices are $i = \overline{0, p(n)}$, $j = \overline{-p(n), p(n)+1}$, $k = \overline{0, r}$, $l = \overline{0, \text{`?'}}$.

Six groups of conjuncts.

C1. Assures that machine is in one state at a moment.

i. $(Q_{i,0} \vee \ldots \vee Q_{i,r})$ where i

ii. $\overline{Q_{i,k}} \vee \overline{Q_{i,k'}}$ where $\forall i \forall k, k' : 0 \leqslant k < k' \leqslant r$

C2. Assures that position of head is determined.

i. $(H_{i,-p(n)} \vee \ldots \vee H_{i,p(n)+1})$

ii. $\overline{H_{i,j_1}} \vee \overline{H_{i,j_2}}$

C3. Assures only one symbol in a position.

i. $(S_{i,j,0} \vee \ldots \vee S_{i,j,r})$

ii. $\overline{S_{i,j,k}} \vee \overline{S_{i,j,k'}}$

C4. Sets initial configuration.

i. $S_{0,1,l_1}, \ldots, S_{0,n,l_n}$

ii. $S_{0,1,0}, S_{0,n+1,0}, \ldots, S_{0,p(n)+1,0}$

iii. $Q_{0,0}$

iv. $H_{0,1}$

C5. Makes sure accepting state is reached.

$Q_{p(n),1}$

C6. Garantees that steps are performed correctly. [1]

i. $(H_{i,j} \vee \overline{S_{i,j,l}}) \vee S_{i+1,j,l}$ – positions that are not observed are not changed.

ii. $\forall (q_{k'}, s_{l'}, \Delta) = \delta(q_k, s_l) \begin{cases} (\overline{H_{ik}} \vee \overline{H_{ij}} \vee \overline{S_{ijl}}) \vee Q_{i+1,k'} \\ (\overline{H_{ik}} \vee \overline{H_{ij}} \vee \overline{S_{ijl}}) \vee Q_{i+1,l'} \\ (\overline{H_{ik}} \vee \overline{H_{ij}} \vee \overline{S_{ijl}}) \vee Q_{i+1,j+\Delta} \end{cases}$

iii. If $q_k \in \{q_Y, q_N\}$ use the conjunct given above, but let $k' = k$ and $l' = l$.

■

Theorem 2 *3-SAT in in NPC*

Proof of this theorem can be found elsewhere.

Theorem 3 *Vertex Cover, Independent Set and Clique are in NPC.*

Proof. Let us note that the following statements are equivalent

- V' is vertex cover.

[1] Recall that $x{>}y$ can be written $!x \vee y$. Left part of implication will be paranthesised below

- $V \setminus V'$ is independent set
- $V \setminus V'$ is a clique in the complement of (V, A)

Thus, it's enough to prove NP-completeness of one of these problems, for instance, independent set. It is done by reduction from 3-SAT. For $E = E_1 \dots E_n$ build a graph by the following rules:

- For each variable x_i, make two vertices x_i and $\overline{x_i}$. Add an edge between them.
- For each conjunct E_k, make vertices corresponding to each of the three literals in the conjunct. Link all three vertices together, making an triangle
- Literal is either x_i or $\overline{x_i}$. Vertex for the literal should be joined with vertex corresponding to either x_i or $\overline{x_i}$, accordingly.

The number of independent vertices to search is $n+p$, where p is the number of variables. **Note that numbers of independent vertices cannot exceed $n + p$.**

If E is satisfiable, independent set can be constructed thusly.

- For all variables, append vertex x_i to the independent set, if $x_i = false$ and append $\overline{x_i}$ otherwise.
- For each E_k some of it's literals is true. Since by construction variable vertex, joined with that literal is not included in the independent set, the literal can be added.

This way, independent set of cardinality $n + p$ is created.

Proof in the other direction is similar. Reader is expected to devise it in much less time than required to typeset it. ■

Theorem 4 *Hamiltonian circuit is in NPC.*

The proof requires a lot of drawing...

1.2 More NP-complete problems

Problem 9 (Interrupt-free scheduling) *Given a number of tasks N, theirs durations τ_i, number of processors m and time limit T, determine, if there's exists an interrupt-free schedule not exceeding the specified time*

Theorem 5 *Interrupt-free scheduling is in NPC.*

Proof. Partition is reducible to scheduling. For the set A create $N = |A|$ tasks with $\tau_i = a_i$, let $m = 2$, and $T = \frac{1}{2} \sum a_i$

Problem 10 (Ordering within interval) *Given a number of tasks N, theirs durations τ_i, allowed time ranges (b_i, f_i) and time limit T, determine if there's exists an interrupt-free schedule on one processor not exceeding the specified time.*

Theorem 6 *Ordering within interval is in NPC.*

Proof. Partition is reducible to ordering within interval. For the set A create $N = |A|$ tasks with $\tau_i = a_i$, let $m = 2$ and $T = 1 + \frac{1}{2}\sum a_i$. Create an auxiliary task with duration of 1 and fix in in the middle by setting time range to $(B/2, B/2 + 1)$. For all the other tasks set time range to $(0, T)$ ■

Problem 11 (Hitting set) *Given a set S and a collection $C_1, \ldots, C_n$, $C_i \subseteq S$ and number K, is there exist $S' \subseteq S : |S'| < K, \forall C_i \exists x \in S' : x \in C_i$*

Theorem 7 *Hitting set is in NPC.*

Proof. Use vertex cover. For graph (V,A), let $S = V$ and $\forall (u, v) \in A$ append set $\{u, v\}$ to the collection. ■

Problem 12 (Subgraph isomorphism) *Find, if graph G_1 contains subgraph, isomorphic to graph G_2*

Take complete graph with K vertices as G_2, and see that clique is reducible to subgraph isomorphism. ■

Problem 13 (Bounded degree spanning tree) *Given a graph $G = (V, A)$ and a number K find if there exists spanning tree such that degree of every vertex if less than K*

Theorem 8 *Bounded degree spanning tree is in NPC.*

Proof. Hamiltonian path is actually a spanning tree with degree less than two (or one)[2] ■

Problem 14 (Knapsack) *Having $n$objects with volumes v_i and prices s_i, and a knapsack of volume V, can we select some objects (find $N' \subseteq \{1, \ldots, n\}$) so that they can be placed ($\sum_{i \in N'} v_i < V$) yet their price is more that some given limit S, ($\sum_{i \in N'} x_i > S$).*

Theorem 9 *Guess what... Those who will can demand a $123.3i$ award* ☺

Proof. Partition is reducible to knapsack. For a set A create $|A|$ objects set $v_i = a_i$, $s_i = a_i$, $V = S = \sum(a_i \in A)/2$ ■

Problem 15 (Longest path in graph) *Is there simple path in graph (V, A) passing through more than K vertices?*

Cf. Hamiltonian circuit. **Note.** For oriented graph without circuits this problem is polynomial. (Who will doubt?)

Problem 16 (Largest common subgraph) *Find, is graphs G_1 and G_2 have some isomorphic subgraphs with more that K vertices.*

[2]It depends on whether the graph is directed or not

Subgraph isomorphism trivially reduces to this problem.

Problem 17 (Minimum Sum of Squares) *Given a set $(a_1, \ldots, a_n)$ and integers J, K, can A be partitioned into K disjoint sets so that $\sum_{i=1}^{K}(\sum A_k)^2 \leqslant J$*

Reduction is from partition. Let $K = 2$ and $J = \frac{B^2}{2}$. If partition is possible, then $(\frac{B}{2})^2 + (\frac{B}{2})^2 = \frac{B^2}{2} = J$. If partitions is not possible then for every A_1 and A_2 $\sum A_1 = \frac{B}{2} - \epsilon$ where $\epsilon > 0$. The sum them will be $\frac{B^2}{2} + 2 * \epsilon^2$

Problem 18 (Late tasks weight minimization) *Given set of N tasks, with executions times τ_i, allowed ranges $[b_i, f_i]$ and weights i, is it possible to create schedule for one processors so that sum of weights of all late tasks is less that K?*

Reduction is from partition. Let $\tau_i = w_i = a_i$, $b_i = 0$, $f_i = B/2$, $K = B/2$.

Problem 19 (Bin packing) *Having an infinite number of bins, each of volume V, can we put set of N objects with volumes v_i into less than K bins.*

This problem is extremely similar to scheduling. The only difference seems to be in variable names.

Problem 20 (Cosine product integral) *Given numbers $a_1, \ldots, a_n$, is it true that $\int_0^{2\pi}[\cos(a_i x)]dx \neq 0$.*

1. $$\int_0^{2\pi} \cos(ax)dx = \begin{cases} 2\pi & \text{if } a = 0 \\ 0 & \text{otherwise} \end{cases}$$

2. $$cos(a_1 x)cos(a_2 x) = \frac{cos(a_1 + a_2)x + cos(a_1 - a_2)x}{2}$$

   $$cos(a_1 x)cos(a_2 x)cos(a_3 x) = \frac{\begin{array}{c} cos(a_1 + a_2 + a_3)x + cos(a_1 + a_2 - a_3)x + \\ +cos(a_1 - a_2 + a_3)x + cos(a_1 - a_2 - a_3)x \end{array}}{4}$$

3. From the formulae above, we conclude that product of cosines is the sum of 2^{n-1} cosines, and coefficients in those cosines correspond to every possible partition of $\{a_1, \ldots, a_n\}$. So, $\int \neq 0$ iff exists coefficient equal to zero, i.e. iff answer to partition problem is "yes".

Problem 21 (Dominating set) *Dominating set in graph $G = (V, A)$ is $V' \subseteq V : \forall v \in V \setminus V' \exists u \in V' : (u, v) \in A$ The question is whether is a given graph exists dominating set with size less than K.*

Reduction from vertex cover. Confine discussion to graphs without isolated vertices. Note that for such graphs *any* vertex cover is also a dominating set. The opposite is false: consider a triangle: any signle vertex is dominating set but not vertex cover. It is possible to force at least one end vertex of each edge to be included to dominating set.

Graph G' is made by creating, for each edge, an auxillary vertex and joining it with both endpoints. Vertex cover in original graph will be dominating set in G' still. On the other side, every dominating set should contain, for each edge, either one of it's endpoints or the auxillary vertex for the edge. If the former case apply to all the edges, we have vertex cover in original graph. Otherwise, it is possible to remove auxillary vertex from dominating set and add one of the endpoints to it, still having dominating set.

Problem 22 (Ordering with minimum delay(?)) *Given set of N tasks, with executions times τ_i and allowed ranges $[b_i, f_i]$, is it possible to create schedule for one processor so that number of late tasks is less that K?*

Unlike with late tasks weight minimization, reduction from partition is not possible directly. We could try to create τ_i tasks for original task i, but such a reduction wouldn't be polynomial.

Take clique problem with graph $G = (V, E)$ and number J. Let's use the obvious fact that if we have $J(J-1)/2$ edges, the number of theirs endpoints is not less than J and equal only in the case of a complete graph. New problem is: $|V| = n, |E| = m$

i. $N = V \cap E$.

ii. $K = m - \frac{J(J-1)}{2}$.

iii. $\forall i \in V f_i = n + m$.

iv. $\forall i \in E f_i = \frac{J(J+1)}{2}$.

v. $\forall (i,j) \in E\ i \prec (i,j)\ j \prec (i,j)$.

So, it is required that at least $\frac{J(J-1)}{2}$ edges are scheduled in first $\frac{J(J+1)}{2}$ time slots. It is possible iff exists clique of J vertices. In this case we can schedule all those nodes and then edges.

2 Pseudopolynomial algorithms

Consider problem Π. For each $I \in \Pi$, in addition to length function $l(I)$, define maximum function $M(I)$. Two pairs (l_1, M_1) and (l_2, M_2) are said to be polynomially eqivalent, if two condition hold:

1. $\exists p_1, p_2 : l_1(I) \leqslant p_2(l_2(I)),\ l_2(I) \leqslant p_1(l_1(I))$
2. $\exists q_1, q_2 : M_1(I) \leqslant q_2(M_2(I)),\ M_2(I) \leqslant q_1(M_1(I))$

Algorithm A is pseudo-polynomial if $\exists p$ such that complexity is bounded by $p(l(I), M(I))$.

Problem Π is problem with numeric parameters if $\nexists p : M(I) \leqslant p(l(I))$, $\forall I \in \Pi$.

Theorem 10 *Unless $P = NP$, no pseudopolynomial algorithm can exist for problem without numeric parameters.*

2.1 Algorithm for Partition

Here a pseudopolynomial algorithm for partition problem is presented. Input is the set $\{a_1, \ldots, a_n\}$. Let $B = \sum a_i/2$. The algorithm creates a matrix T, with elements defined thusly

$$t_{ij} = \begin{cases} 1 & \text{if } \exists \overline{N} \subseteq \{1, \ldots, n\} : \sum_{k \in \overline{N}} a_k = j \\ 0 & \text{otherwise} \end{cases}$$

The first row is very simple to create: for each a_i set $t_{0,a_i} = 1$. If row $i-1$ is already created, row i is created using these rules.

1. $\forall j : t_{i-1,j} = 1 \quad t_{ij} \leftarrow 1$. (If we could form sum S from $i-1$ first elements, this sum can be with equal success formed from first i elements.)
2. $\forall j : t_{i-1,j} = 1 \quad t_{i,j+a_i} \leftarrow 1$.

When the matrix is constructed, one should only look at $t_{n,B/2}$. Complexity is $O(nB/2)$.

2.2 Algorithm for Knapsack

You can devise algorithm yourself.

2.3 Algorithm for Scheduling

Long...

3 Strong NP-completeness

Π' is subproblem of Π if $\mathcal{D}_{\Pi'} \subseteq \mathcal{D}_{\Pi}$ and $Y_{\Pi'} = \mathcal{D}_{\Pi'} \cap Y_{\Pi}$.
Π_p is polynomial (?) of Π, if Π_p is subproblem Π and $\forall I \in \Pi_p \; M(I) \leqslant p(l(I))$
Π is said to be strong NP-complete if $\Pi \in NP$ and $\forall p \Pi_p \in NPC$
(...)

Definition 1 (Pseudo polynomial reduction) $\Pi \leqslant_{pp} \Pi'$ *if* $\exists f : \mathcal{D}_{\Pi} \rightarrow \mathcal{D}_{\Pi'}$*:*

1. $\forall I \in \mathcal{D}_{\Pi} \; I \in Y_{\Pi} \iff f(I) \in Y_{\Pi'}$
2. *f can be computed by pseudopolynomial algorithm.*

3. $\exists p_1 : p_1(l'(f(I))) \geqslant l(I)$
4. $\exists p_2 : M'(f(I)) \leqslant p_2(l(I), M(I))$

Theorem 11 *If* Π *is strong NP,* $\Pi' \in NP$ *and* $\Pi \leqslant_p p\Pi'$*, then* Π' *is strong NP.*

Proof. Take arbitrary polynomial p. According to (?) $\Pi_p \in NP$. Then

$$(4)M'(f(I)) \leqslant p_2(l(I), M(I)) \leqslant p_2(l(I), p(l(I))) \leqslant \langle(3)\rangle \leqslant p_2(p_1(l'(f(I))), p(p_1(l'(f(I))))) = \overline{p}(l'(f(I)))$$

So, Π_p is reduced to polynomial reduction of Π'. Reduction is itself polynomial: its complexity is less than $r(l(I), M(I)) \leqslant r(l(I), p(l(I))) = \overline{r}(l(I))$. So, if $\Pi_{\overline{p}}$ is in NP. (It should be proved that arbitrary $\overline{p}$ can be made by proper selection of p.

4 Turing reducibility

Definition 2 (Turing reducibility) $\Pi_1 \leqslant_t \Pi_2$ *iff* $\exists A_1$*, which solves* Π_1*, using algorithm* A_2 *for solving* Π_2*, and if* A_2 *– polynomial, then* A_1 *– polynomial too.*

Definition 3 (Class NPH) $NPH = \{\Pi : \exists \Pi' \in NPC, \Pi' \leqslant_t \Pi\}$

5 Methods of solving NPC problems

5.1 Approximate algorithms

5.2 Search algorithms

5.3 Randomized algorithms

Some facts from probability theory:

I. Conditional probabily of event A on condition that event B occured is $P(A|B) = \frac{P(AB)}{P(B)}$. It is defined only if $P(B) \neq 0$.

II. Total probability. Let A, $B_1, \ldots, B_n$ be events. If $P(B_iB_j) = 0, i \neq j$ and $P(\cup_i B_i) = 1$, then $P(A) = P(A|B_1)P(B_1) + \ldots + P(A|B_n)P(B_n)$.

Consider a problem: find is $g(x_1, \ldots, x_n) \stackrel{?}{\equiv} h(x_1, \ldots, x_n)$ Before presenting randomized algorithm for the problem, we need one fact.

Lemma 1 (Somebody's lemma) *If* $f(x_1, \ldots, x_n)$ *with degree of any variable* $\leqslant k$*,* $f \not\equiv 0$*,* $\xi_1, \ldots, \xi_n$ *– independent random variables with uniformly distributed over integer values in range* $[0, N-1]$*, for some* N *then* $P(f(\xi_1, \ldots, \xi_n) = 0) \leqslant \frac{kn}{N}$

Proof. Use induction on the number of variables.

a) $n = 1$. Polynomial has no more than k roots. If all those roots are integers in the range $[0, N-1]$, $P(f(\xi_1) = 0) = \frac{k}{N}$. If roots are not such luckily placed, probability will be even less.

b) $n-1 \to n$. Use this repsentation $f(x_1, \ldots, x_n) = f_0(x_2, \ldots, x_n) + f_1(x_1, \ldots, x_n)x_1 + \ldots + f_t(x_2, \ldots, x_n)x_1^t$, $t \leqslant k$. Note that $f_t(\xi_2, \ldots, \xi_n) \neq 0$. Using total probability formulae, we can write

$$\begin{aligned} P(f(\xi_1, \ldots, \xi_n)) &= \overbrace{P(f()|f_t(\xi_2, \ldots, \xi_n) = 0)}^{\leqslant 1} P(f_t(\xi_2, \ldots, \xi_n) = 0) + \\ &+ P(f()|f_t(\xi_2, \ldots, \xi_n) \neq 0) \overbrace{P(f_t(\xi_2, \ldots, \xi_n) \neq 0)}^{\leqslant 1} \leqslant \\ \leqslant P(f_t(\xi_2, \ldots, \xi_n) = 0) &+ P(f(\xi_1, \ldots, \xi_n) = 0 | f_t(\xi_2, \ldots, \xi_n) \neq 0) \leqslant \\ &\leqslant \frac{k(n-1)}{N} + \frac{k}{N} = \frac{kn}{N} \end{aligned}$$

We are ready to describe the algorithm for polynomials equvivalence checking. The question can be formulated as $f \stackrel{?}{\equiv} 0$. Assume that functions f and g can be effectively computed. Generate random integer values uniformly distibuted in range $[0, 2kn-1]$ and compute the value of the function in that random point. If it is not zero, result is computed. Otherwise make another try, etc. If $f \not\equiv 0$, then, according to lemma, $P(\overline{\xi}) = 0) \leqslant \frac{kn}{2kn} = \frac{1}{2}$. Since all random values are independent, probability to get M zeroes when $f \not\equiv 0$ is less than $(\frac{1}{2})^M$. With sufficiently large M, $f \equiv 0$ is very probable.

Another illustation is 2DM-matching(?). Construct matrix M

$$m_{ij} = \begin{cases} x_{ij} & \text{if } (a_i, b_j) \in A \\ 0 & \text{otherwise} \end{cases}$$

St. Full 2DM exists iff $det(M) \not\equiv 0$.
Proof. If $det(M) \equiv 0$ it means that every $(\ldots)$ contains zero. In fact, if any $()$ has no zero, then by increasing at the same time every variable in $()$, we can make absolute value of determinant arbitrary large. If every $()$ contains zero, then 2DM is not possible, since for every possible correspondence, presence of zero means that some pair can't be joined because of absence of an edge.

Determinant is a polynimial. One can compute if effectively using Gaussian elimination. So, previous algorithm can be applied in this case.

6 Additional proofs

Here we will establish NP-completeness for tasks that were not covered in the lecture course.

6.1 3DM

6.2 Partition

The proof can be found at [some URL]

6.3 Graph coloring

6.4 Hamiltonian circuit